ATTACHMENT G

JÜLICH
FORSCHUNGSZENTRUM

Accelerator Area Network

Application Programming Interface (API)

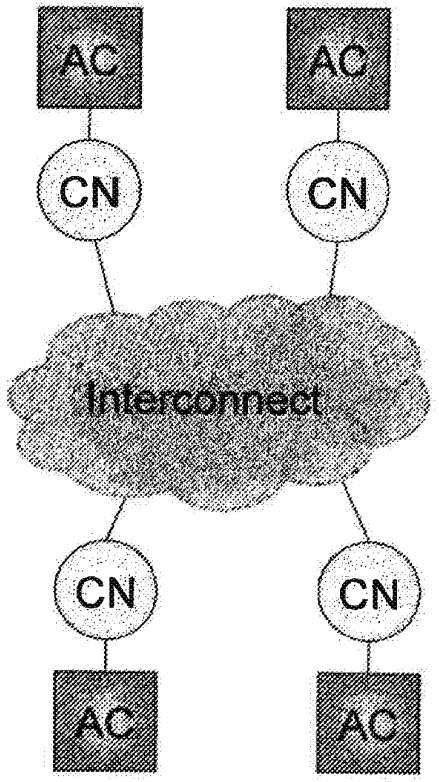
24. September 2010 | Sebastian Rinke Daniel Becker Felix Wolf

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
FORSCHUNGSZENTRUM

Accelerators in Cluster Environments

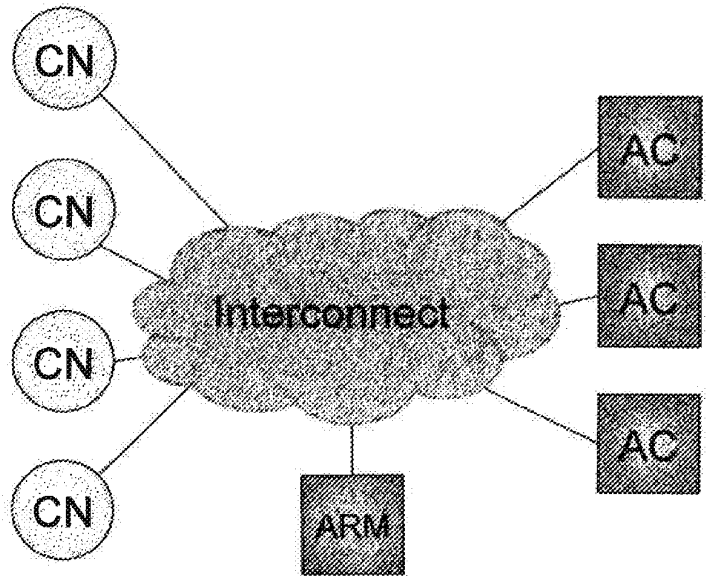
- Current state:
  - Tight coupling of accelerators (AC) to compute nodes (CN)
  - High bandwidth / low latency between compute node and accelerator
  - User has simple view of system
- Limitations:
  - Static AC-to-CN assignment
  - Over-/undersubscription of accelerator based on application
  - No fault tolerance in case of accelerator failure



JÜLICH
FORSCHUNGSZENTRUM

Accelerator Area Network: Basic Idea

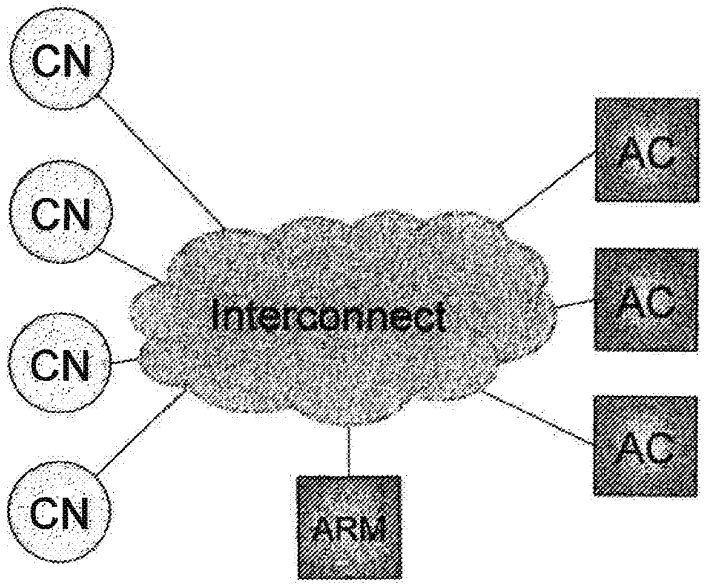
- Loose coupling of accelerators (AC) to compute nodes (CN)
- Sharing of accelerators between compute nodes
- AC-to-CN assignment by accelerator resource manager (ARM):
  - Static (at job start)
  - Dynamic (at runtime)
- Accelerator communication through network protocol



JÜLICH
FORSCHUNGSZENTRUM

Objectives

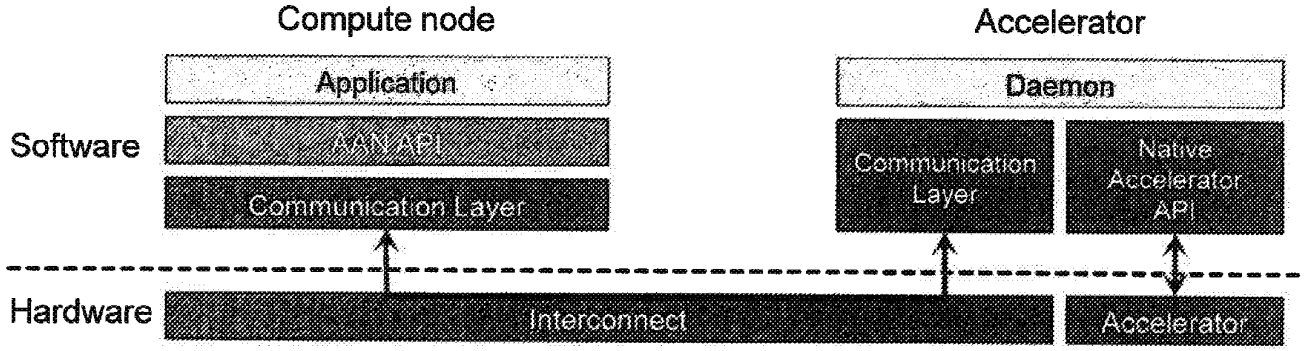
- Dynamic and static AC-to-CN assignment
- Accelerator allocation on application needs
- Fault tolerance in case of accelerator failure
- Support for incremental system development
- All compute nodes share same growth capacity



JÜLICH
FORSCHUNGSZENTRUM

Application Programming Interface (API)

- Compute nodes *transparently* control remote accelerators through API function calls
- API *abstracts* and *enhances* actual native programming model of accelerators
- API provides means for *fault tolerance* in case of an accelerator failure
- Communication protocol involved in API calls layered on top of a communication layer such as MPI:



JÜLICH
FORSCHUNGSZENTRUM

Application Programming Interface (API)

- Current set of API calls:
  - **aanInit(*accelerator*)**
    Initializes *accelerator* before use
  - **aanFinalize(*accelerator)***
    Releases bookkeeping information on *accelerator* after use
  - **aanMemAlloc(*address, size, accelerator*)**
    Allocates *size* Bytes of memory on *accelerator*
    Returns *address* of allocated device memory
  - **aanMemFree(*address, accelerator*)**
    Releases the memory starting at *address* on *accelerator*
  - **aanMemCpy(*dst, src, size, direction, accelerator*)**
    Copies *size* Bytes from *src* to *dst* memory address
    The *direction* of the copy operation can be:
    (i) *accelerator* to host,
    (ii) host to *accelerator*

JÜLICH
FORSCHUNGSZENTRUM

Application Programming Interface (API)

- **aanKernelCreate(*file_name, funct_name, kernel, accelerator*)**
  Creates a kernel defined by the name of the file (*file_name*) and the name of the function (*funct_name*) for execution on *accelerator*
  Returns handle to *kernel*
- **aanKernelSetArg(*kernel, index, size, align, value*)**
  Defines argument for *kernel* execution by its *index* in the argument list, *size*, alignment requirement (*align*), and *value*
- **aanKernelRun(*kernel, grid_dim, block_dim*)**
  Starts kernel execution on the accelerator associated with *kernel* in a previous call to acKernelCreate(). The number of threads is determined by number of threads per block (*block_dim*) and number of blocks in the grid (*grid_dim*)
- **aanKernelFree(*kernel*)**
  Releases the resources associated with *kernel*